Microscope: Mobile Service Traffic Decomposition for Network Slicing as a Service

Chaoyun Zhang, Marco Fiore, Cezary Ziemlicki, Paul Patras

Microscope: Mobile Service Traffic Decomposition for Network Slicing as a Service

Chaoyun Zhang
University of Edinburgh, UK
chaoyun.zhang@ed.ac.uk

Marco Fiore
IMDEA Networks Institute
marco.fiore@imdea.org

Cezary Ziemlicki
Orange Labs
cezary.ziemlicki@orange.com

Paul Patras
University of Edinburgh, UK
paul.patras@ed.ac.uk

ABSTRACT

The growing diversification of mobile services imposes requirements on network performance that are ever more stringent and heterogeneous. Network slicing aligns mobile network operation to this context, by enabling operators to isolate and customize network resources on a *per-service* basis. A key input for provisioning resources to slices is real-time information about the traffic demands generated by individual services. Acquiring such knowledge is however challenging, as legacy approaches based on in-depth inspection of traffic streams have high computational costs, which inflate with the widening adoption of encryption over data and control traffic. In this paper, we present a new approach to service-level demand estimation for slicing, which hinges on *decomposition*, *i.e.*, the inference of *per-service* demands from traffic *aggregates*. By operating on total traffic volumes only, our approach overcomes the complexity and limitations of legacy traffic classification techniques, and provides a suitable input to recent 'Network Slice as a Service' (NSaaS) models. We implement decomposition through Microscope, a novel framework that uses deep learning to infer individual service demands from complex spatiotemporal features hidden in traffic aggregates. Microscope (*i*) transforms traffic data collected in irregular radio access deployments in a format suitable for convolutional learning, and (*ii*) can accommodate a variety of neural network architectures, including original 3D Deformable Convolutional Neural Networks (3D-DefCNNs) that we explicitly design for decomposition. Experiments with measurement data collected in an operational network demonstrate that Microscope accurately estimates per-service traffic demands with relative errors below 1.2%. Further, tests in practical NSaaS management use cases show that resource allocations informed by decomposition yield affordable costs for the mobile network operator.

CCS CONCEPTS

• **Networks** → *Network monitoring*; *Network management*; *Mobile networks*; • **Computing methodologies** → *Artificial intelligence.*

Conference'17, July 2017, Washington, DC, USA

KEYWORDS

Mobile network data traffic, Network slicing, Service demand estimation, Traffic decomposition, Deep learning, Neural networks.

ACM Reference Format:
Chaoyun Zhang, Marco Fiore, Cezary Ziemlicki, and Paul Patras. 2020. Microscope: Mobile Service Traffic Decomposition for Network Slicing as a Service. In *The 26th Annual International Conference on Mobile Computing and Networking (MobiCom '20), September 21–25, 2020, London, United Kingdom.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3372224.3419195

1 INTRODUCTION

Next-generation mobile networks are expected to become a dominant General Purpose Technology (GPT) and enable new services with a trillion-dollar economic output [24], by fulfilling a growing variety of Quality of Service (QoS) needs that range from extreme mobile broadband (eMBB) for, *e.g.*, ultra-high-definition streaming, to ultra-reliable low-latency communication (uRLLC) for, *e.g.*, autonomous driving. In fact, strong service differentiation necessities are already emerging in current deployments, where, *e.g.*, live video streaming must coexist with on-line gaming or shared cloud services. An important instrument for mobile communication infrastructures to answer such diverse necessities is the flexibility in resource management granted by the increasing virtualization of network functions, including dynamic spectrum allocation [7], baseband processing [39], scheduling [3], or task containerization [54].

Network slicing and MANO. On top of these technologies, operators are foreseen to deploy *network slicing*, by isolating dedicated resources and providing customized logical instances of the physical infrastructure to each tenant [20]. Under emerging 'Network Slice as a Service' (NSaaS) models, tenants will obtain full control of the resources and functions allocated within the slices they hold. NSaaS will allow tenants to take advantage of their precise knowledge of end-to-end service performance (*e.g.*, application-level Quality of Experience indicators for individual users) to drive fine-grained network slice configurations (*e.g.*, flow-level traffic engineering) [1, 9]. In such scenarios, network operators remain in charge of performing the management and orchestration (MANO) of resources dedicated to each slice [12]. Critical to MANO is the anticipatory provisioning of isolated capacity (*e.g.*, spectrum, computation, storage, or transport) to each slice. This requires knowledge of the *total traffic demand generated by each mobile service* at a time granularity of minutes, *i.e.*, the resource reallocation periodicity supported by state-of-the-art Virtual Infrastructure Manager (VIM) and Network Function Virtualization (NFV) architectures [52].

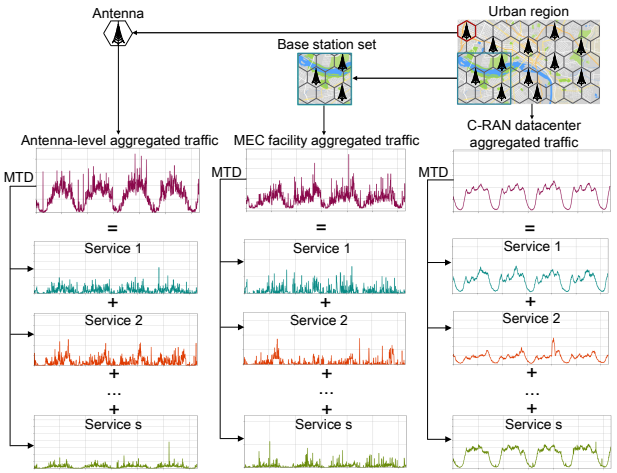

Figure 1: Example of MTD at one antenna (left), a Mobile Edge Computing (MEC) facility controlling a few base stations (center), and a Cloud Radio Access Network (C-RAN) datacenter managing many Distributed Unit (DU) (right). In each scenario, aggregate mobile data traffic (top time series) is decomposed into demands for individual services. Note that the scales of time series are different at each level.

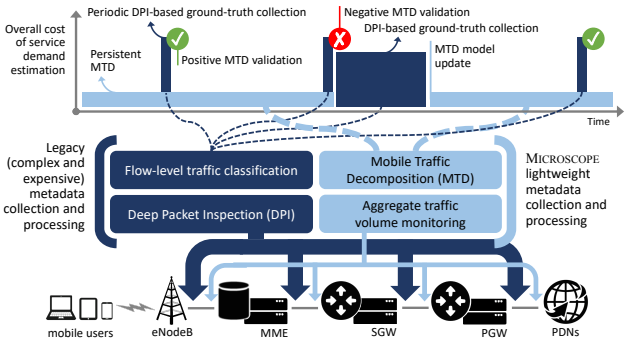

Figure 2: Hybrid service-level demand estimation for NSaaS capacity provisioning. DPI-based and MTD models are combined, so that lightweight decomposition is used as the standard solution, and expensive flow-level classification is only triggered when MTD model retraining is needed.

Traffic classification for NSaaS. However, operators do not have the direct visibility of the service-level traffic required to estimate such demands, and have to resort to inference techniques. Current common practices for extracting this information combine two steps: (*i*) *Deep Packet Inspection* (DPI) to collect packet header metadata, *e.g.*, by sniffing on the GPRS Tunneling Protocol user plane (GTP-U) via probes tapping into the interfaces of the Packet Data Network Gateway (P-GW) in 4G systems [36]; (*ii*) *Flow-level classification* on such metadata to identify the associated service.

Yet, running DPI at line rate and at scale is computationally expensive, while the surge in mobile data traffic and rapid growth of transport link speeds beyond the Terabit-per-second barrier exacerbate the problem. Software-only DPI methods incur substantial latency, as they have to go through a demanding process of fetching each packet from the interface, buffering it, passing it to the CPU, waiting for the operating system task scheduler, and finally parsing the multiple protocol headers. This factually limits packet capture to 70% of the line rate, thereby disregarding a substantial fraction of traffic [42]. On the other hand, recent hardware-based or hybrid DPI solutions can operate close to the line rate [13], yet they come at a very high economic cost for the operator: dedicated FPGA hardware must be deployed at every collection point, and expensive equipment updates are required whenever new classification rules for emerging traffic categories are necessary.

Flow-level service identification on DPI-collected data also yields significant challenges. Due to the increasing adoption of traffic encryption, modern traffic classifiers rely on cleartext hostnames in DNS queries, or revealing fields in the TLS handshake, like the Server Name Indication (SNI) [55]. However, recent proposals for DNS over TLS [22] or encrypted SNI in TLS 1.3 [47] will make classifiers based on DNS and TLS ineffective, and oblige the development of more complex fingerprinting techniques [48]. Not to mention that the design of future flow-level classification solutions will be further challenged by privacy concerns, and will have to abide by emerging strict regulations on personal data protection [40].

Overall, there is a concrete risk that current trends in mobile data usage, network architecture performance, security and privacy will render scalable on-line flow-level classification an even more tangled and resource-intensive problem than it is already today.

Mobile traffic decomposition. In this paper, we propose an alternative approach to demand estimation for sliced network MANO. Abiding by the requirements of NSaaS, we target the inference of *service-level* demands that are required for capacity provisioning to individual slices. Our proposal builds on the novel concept of *mobile traffic decomposition* (MTD), *i.e.*, the process of breaking down aggregate traffic time series into individual service demands, as illustrated by the several examples in Fig. 1.

An accurate MTD can be a *viable complement* to legacy approaches for service-level demand inference in emerging NSaaS network management models. As exemplified in Fig. 2, operators could persistently rely on traffic estimates from MTD to drive service-level capacity provisioning, as it is a lightweight solution based on straightforward monitoring of total traffic volumes. Expensive DPI-based classifiers would only be needed to collect the ground truth needed to validate (and possibly update) the MTD models in presence of evolution in the consumption of mobile applications. Hence, DPI-based solutions would be only run on-demand, sporadically and asynchronously across services, with substantially reduced resource requirements and operation costs.

Contributions. MTD is technically challenging, for multiple reasons: (*i*) the decomposition of a single signal into multiple time series yields inherent ambiguity among a multitude of feasible solutions; (*ii*) helpful complex spatial and temporal correlations exist in mobile traffic [61, 62], but capturing these to resolve the ambiguity is not trivial; (*iii*) traditional decomposition techniques used in other domains, including factorial hidden Markov models [66] or neural networks [65], work on single time series, whereas in our case *multiple input time series* must be concurrently decomposed at different network locations (*e.g.*, diverse edge datacenters).

To tackle these challenges and achieve effective and scalable MTD, we design a dedicated deep learning framework. We rely on deep learning due to its demonstrated effectiveness in discovering knowledge from time series under spatial correlations [8], and in operating on large-scale mobile traffic in real time [64]. This leads to the following contributions:

I. We introduce the original concept of mobile network traffic decomposition (MTD), and prove that it can be a low-cost yet effective solution for the real-time inference of the demands generated by individual mobile services.

II. We propose Microscope, a framework that solves the MTD problem effectively by feeding suitably transformed mobile network traffic to interchangeable deep neural network architectures, including a new class of deformable convolutional neural networks that is explicitly designed for decomposition.

III. We experiment with metropolitan-scale measurement data collected in a production network serving a major European city, and show that Microscope can infer per-service traffic demands with relative errors below 1.2%.

IV. We provide a quantitative analysis of how Microscope would affect practical network operations, showing that resource allocations based on per-service demands estimated via MTD entail costs that are are on par with those incurred when perfect knowledge of per-service traffic is available.

2 MOBILE TRAFFIC DECOMPOSITION

We provide a formal definition of the MTD problem in Sec. 2.1. Then, we outline the high-level structure of the Microscope framework we propose to solve such a problem in Sec. 2.2.

2.1 Problem Formulation

Let us consider a geographical region where mobile network coverage is provided by a set $\mathcal{A}$ of antennas,[1] which accommodate traffic generated by a set $\mathcal{S}$ of mobile services. We denote by $d_a^s(t)$ the traffic demand (expressed in Mbps) accommodated by antenna $a \in \mathcal{A}$ for a specific service $s \in \mathcal{S}$ at time t. The sum of demands over all services gives the aggregate demand at antenna a and time t, $d_a(t) = \sum_{s\in\mathcal{S}} d_a^s(t)$. A mobile network traffic *snapshot* is the set of demands recorded at all antennas in the target region at a specific time. This applies both to individual services, leading to a *service snapshot* $D^s(t) = \left\{d_a^s(t) | a \in \mathcal{A}\right\}$, and to traffic aggregates over all services, obtaining an *aggregate snapshot* $D(t) = \{d_a(t) | a \in \mathcal{A}\}$.

The MTD problem is formally defined as that of inferring the service snapshots $D^s(t)$ of all services $s \in \mathcal{S}$ at current time t, by only knowing the aggregate snapshots up to T previous time instants, *i.e.*, $\{D(t-T+1), \ldots, D(t)\}$. If we denote by $\mathcal{D}^s(t) = \{D^s(t) | s \in \mathcal{S}\}$ the set of current service snapshots, the solution to the MTD problem can be expressed as

$$\widetilde{\mathcal{D}}^s(t) := \underset{\mathcal{D}^s(t)}{\arg\max}\ p\big(\ \mathcal{D}^s(t) \mid \{D(t-T+1), \ldots, D(t)\}\ \big), \quad (1)$$

where $\widetilde{\mathcal{D}}^s(t)$ denotes the estimated current traffic demands disaggregated over the service set $\mathcal{S}$ for all antennas in $\mathcal{A}$, and $p(\cdot)$ is the probability of the argument. The MTD results are restricted by two obvious constraints in the system *i.e.*,

$$\tilde{d}_a^s(t) \geq 0, \qquad \forall a \in \mathcal{A}, \forall s \in \mathcal{S}, \forall t, \quad (2)$$

$$\sum_{s\in\mathcal{S}} \tilde{d}_a^s(t) = d_a(t), \qquad \forall s \in \mathcal{S}, \forall t. \quad (3)$$

The expression in (2) enforces that all estimated traffic demands are positive, while (3) implies that the sum of the per-service traffic must be equal to the aggregate traffic.

We stress that MTD is a fundamentally different problem from prediction, as it assumes knowledge of the aggregate measurement data at the current time instant t, and aims at inferring information relative to that same time instant. In fact, MTD is a one-to-many problem that seeks to decompose one aggregate traffic measurement into the underlying $|\mathcal{S}|$ per-service snapshots: $\mathcal{D}^s(t)$ includes $|\mathcal{S}|$ snapshots, each featuring the same cardinality as $D(t)$. Iteratively solving this problem over time ultimately leads to the reconstruction of per-service demand time series at each antenna from the aggregate traffic, as originally illustrated in Fig. 1.

2.2 Microscope in a Nutshell

Microscope is a novel machine learning framework that is specifically designed for MTD. Here, we provide an overview of the framework, discussing the functionality and integration of the modules it comprises; the following Sec. 3–5 give full details about the implementation of each component.

As shown in Fig. 3, there are three main elements in Microscope. The first is a *traffic snapshot transformation* block, which receives the current aggregate mobile network traffic measurement data $D(t)$ and converts them into a format that is suitable for the following analysis. Specifically, this component limits the spatial distortion of antenna locations as they are fed to the neural network, by solving an opportune association problem. This transformation is critical to generalizing the framework, since it allows Microscope to accommodate any antenna deployment layout with minimum loss of geographical information. Details are in Sec. 3.

The second component implements a *deep neural network model*, whose goal is learning abstract spatiotemporal correlations that are unique of mobile traffic, as needed to solve the MTD problem. To this end, the model takes as input the transformed measurement data corresponding to the aggregate traffic recorded during most recent T time instants. The framework can accommodate different neural network architectures, and we test a number of variants, including a novel *3D deformable convolutional neural network* (3D-DefCNN) specifically designed for decomposition. Details are in Sec. 4.

The third component is concerned with the *loss function* used to drive the learning process. Also in this case, we consider and compare different options with properties that include (*i*) suitable output normalization, (*ii*) preservation of the total traffic across all service demands, and (*iii*) overall accuracy in solving the MTD problem. The final outputs are the estimated service snapshots $\widetilde{\mathcal{D}}^s(t)$ as per (1). Details are in Sec. 5.

3 MOBILE TRAFFIC TRANSFORMATION

Microscope relies on deep neural network models, and can accommodate a variety of architectures. Among those, convolutional neural networks (CNNs) are especially adapted to MTD, as they

[1] An eNodeB may serve users present in one or multiple sectors, each covered by a co-located antenna with a different azimuth. We refer to a single radio front-end unit as one 'antenna' hereafter.

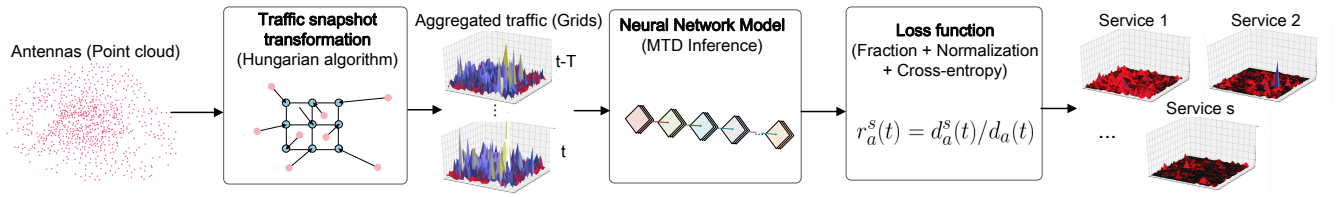

Figure 3: Microscope framework. The pipeline performs transformation of mobile traffic snapshots (left), traffic decomposition via a neural network model (center), and learning driven by a dedicated loss function (right).

can take advantage of spatiotemporal correlations in mobile network traffic to improve the decomposition accuracy. Indeed, previous studies have repeatedly demonstrated the existence of spatial and temporal interrelationships between the demands generated by mobile users [14], which have proven essential in performing data-driven networking tasks. For instance, cell load forecasting is enhanced by harnessing information from adjacent sites, exploiting the fact that the mean of spatial conditional entropy varies with the number of considered adjacent cells [32]. Similarly, traffic super-resolution is improved when taking advantage of long-timescale correlations present in temporal sequences of traffic consumption snapshots [63]. In order to leverage these properties for MTD, the majority of models in Sec. 4 use convolutional entry layers.

However, convolutional layers require an input in matricial form. In our case, the input matrix must describe an aggregate snapshot $D(t)$ of mobile network traffic, as explained in Sec. 2.1. The constraint on the input format creates the problem of mapping antennas to matrix elements. Radio access infrastructure deployments typically have irregular spatial distributions that are driven by the area topography and varied density of subscriber presence, hence are not easily matched to a matrix. This is illustrated in Fig. 4 (left), which portrays the real-world antenna arrangement in the metropolitan-scale network we consider in our experiments. We present our strategy to address this problem in Sec. 3.1, while we discussed its performance and alternative options in Sec. 3.2.

3.1 Minimum Displacement Grid Mapping

We solve the mapping problem by constructing a regular grid that has the same number of points as the number of antennas, and performing a one-to-one antenna-to-point association that minimizes the displacement of the original antenna locations. The rationale for this design is twofold: (*i*) it produces a regular grid, *i.e.*, an inherent matricial form; and, (*ii*) it aims at preserving spatial correlations in traffic that convolutional layers can take advantage of, by linking geographically close antennas to adjacent points of the grid.

3.1.1 Regular Grid Design. To preserve spatial correlations within mobile network traffic, the grid should (*i*) overlap with the coverage of the antennas as much as possible, and (*ii*) help limiting spatial displacements after the mapping procedure. Given the target set $\mathcal{A}$ of antennas, our grid design follows the logic below.

(1) We project the locations of all antennas to an Euclidean space, determine the extreme values on x and y axes, *i.e.*, $x_{\min}, x_{\max}$, $y_{\min}, y_{\max}$, and compute the aspect ratio of the overall coverage region as $\rho = (x_{\max} - x_{\min})/(y_{\max} - y_{\min})$;
(2) We dimension the grid so that it best reflects such an aspect ratio, with $n_r = \sqrt{|\mathcal{A}|/\rho}$ rows and $n_c = n_r/\rho$ columns;

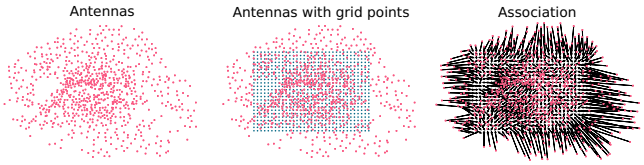

Figure 4: Illustration of the minimum displacement grid mapping in the network deployment used for our experiments. Antennas positions (left) are used to design a grid layout (center). The mapping via the Hungarian algorithm is shown by arrows that denote the displacement from each original antenna position to the assigned grid point (right).

(3) We assign geographical coordinates to the regular grid, by superposing the grid to the antenna positions, and then scaling distances by a factor κ to account for the heterogeneous antenna density typical of urban areas.

The center plot of Fig. 4 portrays the regular grid obtained with the technique above, in the case of the antenna deployment in the left plot of the same figure. In this scenario, we employed $\kappa = 0.25$ after extensive tests, and obtained a 33×24 grid.

3.1.2 Antenna-to-Point Mapping. We aim to minimize the overall displacement when performing the one-to-one association between antennas and grid points. We define the displacement $c_{a,p}$ as the Euclidean distance between the geographical positions of antenna a and grid point p. We construct a cost matrix $C := \{c_{a,p}\}_{|\mathcal{A}|\times|\mathcal{A}|}$ to represent the distances between antennas and grid points. We further define a binary matrix $X := \{x_{a,p}\}_{|\mathcal{A}|\times|\mathcal{A}|}$, where $x_{a,p} = 1$, if and only if the antenna a is assigned to the grid point p. Hence, we formulate the association problem as:

$$\min_{x} \sum_{a\in\mathcal{A}} \sum_{p=1}^{|\mathcal{A}|} c_{a,p} \cdot x_{a,p}, \tag{4}$$

$$\text{s.t.} \sum_{a\in\mathcal{A}} x_{a,p} = 1,\ \forall p \le |\mathcal{A}|; \sum_{p=1}^{|\mathcal{A}|} x_{a,p} = 1,\ \forall a \in \mathcal{A}, \tag{5}$$

where constraints enforce that one antenna will be assigned to only one grid point, and vice versa.

The expressions in (4)–(5) define an assignment problem that is efficiently solved via the Hungarian algorithm [29]. The algorithm has a polynomial complexity $O(|\mathcal{A}|^3)$, hence runs efficiently in practical cases. For the antenna deployment used in our experiments, in Fig. 4, it returns the mapping in the right plot. We observe that only antennas on the outskirts of the urban area are subject to significant spatial shifts, and movements are otherwise small.

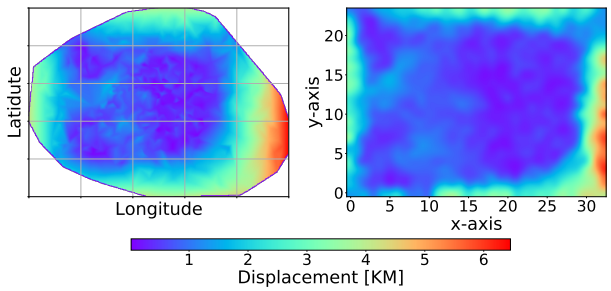

Figure 5: Heatmaps of the spatial displacement incurred by our grid mapping, with respect to the original antenna locations (left), and projected on the regular grid space (right).

3.2 Comparison with Other Mapping Strategies

We provide an in-depth view of the proposed minimum displacement grid mapping's performance in Fig. 5. The heatmaps illustrate the displacement incurred by antennas in the network considered for our experiments, upon traffic transformation. In the left plot, the displacement information is associated to the original antenna positions: this highlights how antennas at the conurbation borders may be shifted by 2 to 6 km, while displacements are below 1 km otherwise. The right plot associates displacement values to the final grid points, and offers an even clearer outlook of the mapping quality: the vast majority of antennas undergoes shifts below 500 m, hence preserving substantial spatial relationships in the traffic.

Although it works reasonably well, our transformation is not the only possible approach to the generation of input traffic matrices. An alternative design could replace the regular grid tessellation of space with a different one, *e.g.*, based on the triangular or hexagonal tiling that is traditionally adopted in ideal network deployment models. Such tessellations, however, are not immediately translated to a matricial form, and introduce one step in the transformation process, thus incurring in higher displacement and weakened spatial correlations. For instance, we experiment with a Voronoi tessellation commonly adopted to mimic the coverage areas of antenna sites [16, 44]. By this, we move antennas to the barycenter of their corresponding Voronoi polygon, and then employ the approach in Sec. 3.1.2. This design yields a very similar traffic matrix to that obtained with our method: 88% of the antennas are mapped to the exact same grid point, and differing elements are 1.66 cells apart (*i.e.*, shifted to a neighboring grid point) on average. Still, the Voronoi design above leads to an average displacement of antenna original locations increased to 1.29 km from 1.22 km in our transformation.

A different solution to reconciling real-world antenna deployments with the input requirement of CNNs could be spatial supersampling [4]. This involves approximating the coverage area of each antenna (*e.g.*, via the Voronoi tessellation above), assuming the traffic recorded at each antenna to be uniformly distributed within the associated coverage area, and superposing a dense grid to the resulting continuous traffic map. Then, each matrix element can be easily filled with the traffic in the underlying map. However, this option has significant shortcomings, since (*i*) it introduces strong unrealistic assumptions about the coverage and spatial distribution of users, and (*ii*) in the case of dense grids, it artificially increases the size of the matrices and the computational cost of the CNN.

Table 1: Legacy neural networks configuration.

Class	Configuration
MLP	Legacy MLP, with 5 hidden layers, and 1,000 hidden units per layer
CNN	Fig. 8, without the 3D convolutional block
LSTM	Legacy LSTM, with 3 stacks, and 500 units per stack
ConvLSTM	Legacy ConvLSTM, with 3 stacks, and 108 channels per stack
ZipNet	Fig. 8, with the 3D deformable convolutional layers in the first block replaced by legacy 3D convolutional layers
DefCNN	Same as ZipNet, with the convolutional layers in the last block replaced by 2D deformable convolutional layers

4 DEEP NEURAL NETWORK MODELS

The core of Microscope is a deep learning architecture. The framework is flexible, and can accommodate a variety of neural network models. We experiment with a number of different architectures proposed in the machine learning literature, which are summarized in Tab. 1, and detailed in Sec. 4.1. In addition, we design a novel *3D deformable convolutional neural network* (3D-DefCNN) that is tailored to MTD in especially complex scenarios: it compensates for the spatial displacement in network traffic snapshots, discovers spatiotemporal correlations in aggregate traffic, and exploits them for decomposition, as thoroughly discussed in Sec. 4.2.

We remark that the openness to various deep learning architectures is an important feature of the Microscope framework. Indeed, our performance evaluation results, presented later in Sec. 6, indicate that there is not a single neural network that works best for all decomposition tasks, but different models should be adopted depending on the target network management scenario.

4.1 Legacy Architectures

We consider a comprehensive set of five legacy deep learning models among the many options available in the vast machine learning literature. (*i*) The multi-layer perceptron (MLP) is the simplest neural network class [17] and we consider it as a baseline solution. (*ii*) Convolutional neural networks (CNNs) are commonly used for imaging applications [28]. (*iii*) Long Short-Term Memory (LSTM) is frequently exploited for modelling sequential data [21], *e.g.*, speech signals and natural language. (*iv*) Convolutional LSTM (ConvLSTM) is a dedicated model for spatiotemporal data forecasting [59]. (*v*) Deep zipper networks (ZipNets) were originally proposed for mobile traffic super resolution tasks with remarkable results [63].

A sixth variant for the neural network model integrated in Microscope is the (*vi*) deformable convolutional neural network (DefCNN) [10] originally proposed for computer vision application, such as image classification [67], object detection and semantic segmentation [45]. Whilst classic CNNs apply fixed geometric transformations to a 2D space, DefCNN architectures perform deformable convolutions over the same input. This allows flexible transformations that can compensate for distortions in the 2D input space.

As discussed in Sec. 3, we adopt a traffic snapshot that mitigates but cannot completely remove the displacement of antenna locations into a regular grid. Therefore, aggregate snapshots recorded in the 2D geographical space are distorted, with a risk that spatial correlations in the original data are misrepresented in the input matrix. DefCNN can reduce such a risk, by enabling convolutional filters to access any element of the input matrix. These connections are dynamically tailored to the input, and can be learned jointly with

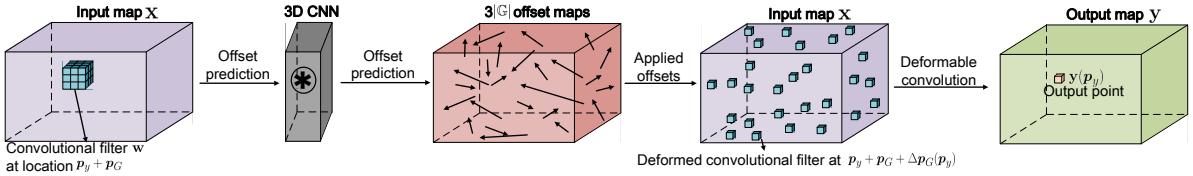

Figure 6: Graphical illustration of 3D deformable convolutional operation.

the other synapses via gradient descent. The approach is equivalent to letting the neural network re-organize the 2D spatial structure of the input data. In our context, this allows identifying and exploiting spatial correlations in the mobile network traffic that the transformation may weaken.

4.2 3D-DefCNN

We propose an additional (*vii*) 3D-DefCNN model, which is an enhancement of the DefCNN. The 3D-DefCNN design stems from an original *3D deformable convolution* operation, which is detailed next, along with the overall network architecture.

4.2.1 3D Deformable Convolution. Legacy DefCNNs perform deformable convolution over 2D input matrices. In MTD, the data fed to the network includes time, thus it is three-dimensional. Specifically, the input consists of the aggregate snapshot set $\{D(t-T+1), \ldots, D(t)\}$ (Sec. 2.1), which is transformed into T subsequent 2D matrices (Sec. 3.1.2). Hence, we extend the deformable convolution operation to the temporal dimension. In doing so, we account for the fact that not all last T input snapshots have the same relevance to MTD at the current time instant, and weight in an adaptive manner the different 2D input matrices.

To achieve our objective, we combine the DefCNN model with 3D convolution, a technique previously used for action recognition [26], which operates over both time and a bidimensional space. Essentially, 3D convolution performs summation over an input map $\mathbf{x}$ weighted by a filter matrix $\mathbf{w}$, which results in an output map $\mathbf{y}$.[2] Each convolutional filter $\mathbf{w}$ has a receptive field $\mathbb{G}$, which defines the spatiotemporal extent of connectivity between different locations in the input. As an example, the receptive field of a $3 \times 3 \times 3$ convolutional filter can be defined as $\mathbb{G} = \{(-1,-1,-1), (-1,-1,0), \ldots, (1,1,0), (1,1,1)\}$. For each location $\boldsymbol{p}_y$ of the output $\mathbf{y}$, the 3D convolution performs the following calculation:

$$\mathbf{y}(\boldsymbol{p}_y) = \sum_{\boldsymbol{p}_G \in \mathbb{G}} \mathbf{w}(\boldsymbol{p}_G) \cdot \mathbf{x}(\boldsymbol{p}_y + \boldsymbol{p}_G), \tag{6}$$

where $\boldsymbol{p}_G$ is a 3D vector in $\mathbb{G}$. For instance, if $\boldsymbol{p}_y = (1,2,3)$ and $\boldsymbol{p}_G = (-1,1,0)$, then $\mathbf{x}(\boldsymbol{p}_y + \boldsymbol{p}_G)$ is the $\mathbf{x}$ value at $(0,3,3)$.

Our proposed 3D-DefCNN extends (6) above with deformability, by adding an offset $\Delta\boldsymbol{p}_G$ to each $\boldsymbol{p}_G$:

$$\mathbf{y}(\boldsymbol{p}_y) = \sum_{\boldsymbol{p}_G \in \mathbb{G}} \mathbf{w}(\boldsymbol{p}_G) \cdot \mathbf{x}(\boldsymbol{p}_y + \boldsymbol{p}_G + \Delta\boldsymbol{p}_G(\boldsymbol{p}_y)). \tag{7}$$

[2]In action recognition tasks, $\mathbf{x}$ and $\mathbf{y}$ are four-dimensional tensors: the first three are the spatiotemporal dimensions; the fourth is the RGB channel dimension. This allows defining dedicated filters on each channel. In our case, we employ a single filter shared across all channels of the input. Our neural network will still produce multiple *channels* throughout the hidden layers, which do not have a direct physical meaning, but rather provide intermediate outputs that aid extracting abstract features. The final layer making predictions has however one output channel for each mobile service.

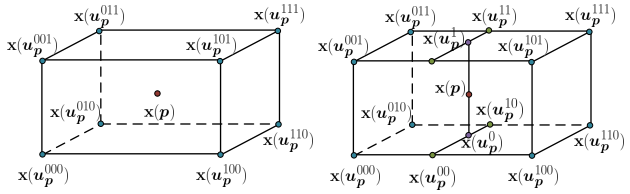

Figure 7: Example of trilinear interpolation, where $\{\mathbf{x}(\boldsymbol{u}_p^{00}), \mathbf{x}(\boldsymbol{u}_p^{01}), \mathbf{x}(\boldsymbol{u}_p^{10}), \mathbf{x}(\boldsymbol{u}_p^{11})\}$ are first computed via linear interpolation, then $\{\mathbf{x}(\boldsymbol{u}_p^{0}), \mathbf{x}(\boldsymbol{u}_p^{1})\}$ are derived, and $\mathbf{x}(\boldsymbol{p})$ is obtained.

Note that: (*i*) for each location $\boldsymbol{p}_y$ a different $\Delta\boldsymbol{p}_G(\boldsymbol{p}_y)$ may be applied; (*ii*) for each location, we have $|\mathbb{G}|$ three-dimensional offsets, which can be globally seen as $3|\mathbb{G}|$ maps of offsets on the input grid, or one $|\mathbb{G}|$ map for each (spatial or temporal) dimension; (*iii*) all $\Delta\boldsymbol{p}_G(\boldsymbol{p}_y)$ offset maps are learned by an additional 3D-CNN layer that takes $\mathbf{x}$ as input.

We illustrate the principle of 3D deformable convolution in Fig. 6. As mentioned above, we first apply a 3D-CNN structure (3 layers) onto the original convolutional filter (which is compact) to predict $3|\mathbb{G}|$ offset maps $\Delta\boldsymbol{p}_G(\boldsymbol{p}_y)$. These offset maps essentially seek to alter the position of the filter elements, in order to scan, at each step, locations in the input that are not necessarily adjacent. The offsets are learned and shared across the different input channels. Subsequently, we apply these offsets to the original convolutional filter, to construct a deformed convolutional filter. Finally, by performing convolution between input and deformed filter, we obtain the output at location $\boldsymbol{p}_y$ via (7). Since 3D deformable convolution operations are fully differentiable, the offsets to be applied can be learned through standard back-propagation. As such, 3D-DefCNNs grant convolutional filters complete freedom to query any location in the input maps. This significantly improves the model flexibility and enables to adapt to any spatial displacement (spatial deformation) and diverse importance of historical data (temporal deformation).

A complication introduced by equation (7) is that the sample positions $\boldsymbol{p} = \boldsymbol{p}_y + \boldsymbol{p}_G + \Delta\boldsymbol{p}_G$ can be fractional. Indeed, while $\boldsymbol{p}_G$ and $\boldsymbol{p}_y$ are indices of $\mathbb{G}$ and $\mathbf{y}$, and thus integer values, $\Delta\boldsymbol{p}_G$ may not be integer, as it is estimated by a CNN. To solve the issue, we compute the deformed input in (7) as:

$$\mathbf{x}(\boldsymbol{p}_y + \boldsymbol{p}_G + \Delta\boldsymbol{p}_G) = \mathbf{x}(\boldsymbol{p}) = \sum_{\boldsymbol{u}_p \in \boldsymbol{U}_p} Tri(\boldsymbol{U}_p, \boldsymbol{p}) \cdot \mathbf{x}(\boldsymbol{u}_p), \tag{8}$$

where $\boldsymbol{U}_p = \{\boldsymbol{u}_p^{000}, \boldsymbol{u}_p^{001}, \ldots, \boldsymbol{u}_p^{111}\}$ are the locations of the eight input samples around $\boldsymbol{p}$, and $Tri(\cdot)$ denotes the trilinear interpolation operator [46], whose principle we show in Fig. 7.

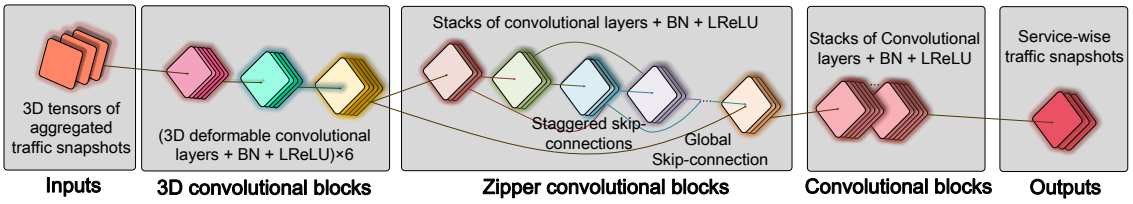

Figure 8: Overall structure of the 3D-DefCNN model, consisting of 3D deformable convolutional blocks (left), zipper convolutional blocks (middle), and standard 2D convolutional blocks (right).

4.2.2 Overall 3D-DefCNN Structure. We embed the 3D deformable convolutional operations above in the complete 3D-DefCNN structure shown in Fig. 8. Our architecture design encompasses three major components: (*i*) 3D deformable convolutional blocks, (*ii*) zipper convolutional blocks, and (*iii*) 2D convolutional blocks.

The 3D deformable convolutional blocks consist of stacks of 3D deformable convolutional layers, batch normalization (BN) layers [25], and leaky rectified linear unit (LReLU) activation layers [34]. As previously detailed, 3D deformable convolutions are employed to mitigate the spatial displacements and perform adaptive weighting over historical observations; in addition, they extract important spatiotemporal patterns in mobile network traffic. BN layers perform normalization over a batch of output of each layer. This effectively reduces output's variance and can significantly accelerate the model training. LReLUs perform as activation functions. They improve the model non-linearity and representability, which enables the model to extract even more abstract features.

The zipper convolutional blocks receive the output of the 3D deformable convolutional blocks and are responsible for feature extraction. The structure of these blocks is inspired by that of deep zipper networks (ZipNets) originally proposed for mobile data traffic super-resolution [63], which work demonstrably well in extracting spatiotemporal correlations hidden in this type of measurement data. More precisely, global and multiple skip connections are employed within these blocks, to perform effective residual learning [19], which is known to make the model more robust by constructing an ensembling system of neural networks with different depths [56]. Skip connections also significantly smoothen the loss surface, which enables faster convergence of the model training [31].

Upon processing by the zipper convolutional blocks, the mobile network traffic data is transformed into highly abstracted representations, ready for the final MTD inference. This is performed by standard 2D convolutional blocks. Compared to the previous blocks, these are configured with a larger number of feature maps, so as to provide sufficient information for the inference process. The last layer of this block has $|\mathcal{S}|$ channels, *i.e.*, feature maps, each corresponding to the decomposed traffic volume of an individual mobile service.

5 DRIVING THE LEARNING PROCESS

We explore three different methods to train the Microscope neural network so that it solves the MTD problem in (1). Namely, we test (*i*) regression, (*ii*) ratio prediction trained with Mean Square Error (MSE), and (*iii*) ratio prediction trained with Cross-Entropy (CE).

(1) The *Regression* method trains neural networks with a loss function over the traffic volume that aims at minimizing the difference between $\widetilde{d}_a^s(t)$ and $d_a^s(t)$, *i.e.*,

$$\mathrm{L}(t) = \frac{1}{|\mathcal{S}| \cdot |\mathcal{A}|} \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} ||\widetilde{d}_a^s(t) - d_a^s(t)||^2. \tag{9}$$

Due to model imperfections, the output obtained with this approach may violate the constraints (2) and (3).

(2) The ratio prediction method based on Mean Square Error (MSE) seeks to infer the fraction of traffic consumed by each service relative to the corresponding aggregate, *i.e.*, $r_a^s(t) = d_a^s(t)/d_a(t)$. Clearly, $\sum_{s \in \mathcal{S}} r_a^s(t) = 1, \forall a \in \mathcal{A}, \forall s \in \mathcal{S}, \forall t$. A softmax function allows an equivalent transformation for the estimated service demand:

$$\tilde{r}_a^s(t) = \frac{\exp(\tilde{\chi}_a^s(t))}{\sum_{s \in \mathcal{S}} \exp(\tilde{\chi}_a^s(t))}. \tag{10}$$

Here $\tilde{\chi}_a^s(t)$ denotes the intermediate (unnormalized) output of the neural network. In essence, by the above we map a vector output with elements of arbitrary values onto a probability vector where the elements are in the (0,1) range [6]. The expression of $\tilde{r}_a^s(t)$ in (10) satisfies both (2) and (3). The network is trained to minimize the MSE between $\tilde{r}_a^s(t)$ and $r_a^s(t)$, with a loss function:

$$\mathrm{MSE}(t) = \frac{1}{|\mathcal{S}| \cdot |\mathcal{A}|} \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} ||\widetilde{r}_a^s(t) - r_a^s(t)||^2. \tag{11}$$

Note that the actual mobile service demand can be easily retrieved from the output ratio as $\tilde{d}_a^s(t) = \tilde{r}_a^s(t) \cdot d_a(t)$. We refer to this approach as *MSE* in the following.

(3) The *cross-entropy (CE)* loss function is formally:

$$\mathrm{CE}(t) = \frac{1}{|\mathcal{A}|} \sum_{a \in \mathcal{A}} \sum_{s \in \mathcal{S}} -r_a^s(t) \log(\tilde{r}_a^s(t)). \tag{12}$$

for the service snapshot estimated at time t. The overall CE for a given time span $\mathcal{T}$ can then be computed as the average over all time instants $t \in \mathcal{T}$. This expression is minimized when the estimated and actual traffic demand ratios match, *i.e.*, $\tilde{r}_a^s(t) = r_a^s(t), \forall a \in \mathcal{A}, \forall s \in \mathcal{S}, \forall t$. This maps to minimizing the *Kullback–Leibler* (KL) divergence between the model and target distribution, under the assumption that the ratio $r_a^s(t)$ follows a multinomial distribution [38]. Training a neural network with CE usually finds a better optimum than other loss functions when the output is normalized [15], as in our case. This will be later confirmed by comparative evaluations.

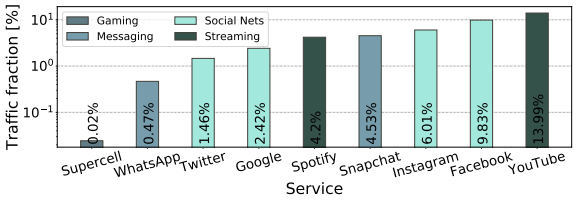

Figure 9: Overview of the traffic generated by services in the set $\mathcal{S}$ considered in our study.

6 EXPERIMENTS

We implement Microscope using the open-source Python libraries TensorFlow [2] and TensorLayer [11]. We train and evaluate the framework on a high-performance computing cluster with two NVIDIA Tesla K40M GPUs with 2280 cores. The model parameters are optimized using the popular Adam stochastic gradient descent-based optimizer [27].

The evaluation is organized as follows. we first present in Sec. 6.1 the reference scenario used to run our experiments, and then introduce in Sec. 6.2 the metrics used to assess the accuracy of Microscope. We perform in Sec. 6.3 a comprehensive comparative evaluation of MTD performance on mobile network traffic recorded at the antenna level, which corresponds to end-to-end NSaaS settings where each network slice enjoys dedicated spectrum [35]. In Sec. 6.4, we investigate how such performance vary at different network levels, including MEC facilities, C-RAN or core network datacenters where network slices are also to be implemented. Finally, we comment on the complexity of the different models used in Microscope, discuss accuracy-complexity trade-offs, and investigate the importance of the temporal deformation operation for the overall system performance in Sec. 6.5.

6.1 Reference Scenario

We conduct our experiments on real-world 3G/4G mobile network traffic data collected by Orange, a major European mobile operator, in a large metropolitan area during 85 consecutive days. Each mobile network traffic snapshot consists of the total demand accumulated over a 5-minute time interval at 792 different antenna sectors, for different mobile services separately.

The measurement data is collected via DPI at the P-GW, and proprietary traffic classifiers are used to associate flows to specific services. Due to confidentiality constraints, we do not disclose the target urban region, or operation details of the classifiers. However, internal performance assessments by the operator report a typical flow-level classification accuracy at around 90%. We remark that all measurements were carried out in compliance with applicable local and European regulations, under the supervision of the competent national privacy agency. In particular, the dataset employed in our study only contains information on mobile service traffic accumulated at the antenna level, and does not hold any personal information about individual subscribers.

The set of services $\mathcal{S}$ considered in our analysis includes mobile games (Clash Royale and Clash of Clans, grouped under the Supercell label), messaging apps (Snapchat and WhatsApp), social media (Facebook, Twitter, Instagram), video (YouTube) and audio (Spotify) streaming platforms, as well as Google services. Fig. 9 illustrates the fraction of total traffic induced by each service. The rationale for selecting these nine services is that they are reasonable candidates for the allocated of dedicated network slices. Indeed, they have clear Quality of Service (QoS) requirements, and are *heavy hitters*, *i.e.,* generate sizeable amounts of network traffic: owing to the well-known Zipfian distribution of the demand across services [36], these services are in fact responsible for more than 42% of the total mobile data traffic in the target region. In addition, the set $\mathcal{S}$ encompasses a variety of application types, whose demands yield strongly dissimilar temporal dynamics [36]; as such, these services provide a challenging but realistic ground for MTD.

The neural network is trained and validated on data collected by the operator in the first 68 days (60%), and tested on the traffic observed during the last 17 days (20%). In these conditions, the training process converges at the tenth epoch, taking around 48 hours in total. Performing MTD inference in the considered large-scale scenario requires less than 1 second per instance.

6.2 Performance Metrics and Benchmarks

We evaluate the performance of Microscope by means of two complementary metrics, *i.e.,* mean absolute error (MAE) and Normalized MAE (NMAE). MAE is a commonly employed measure of prediction accuracy, and is known to be stable and mostly insensitive to large error values [58]. It is formally defined as:

$$\mathrm{MAE}(t) = \frac{1}{|\mathcal{S}| \cdot |\mathcal{A}|} \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} |\widetilde{d}_a^s(t) - d_a^s(t)|. \tag{13}$$

In addition, we explain the relative significance of the error via NMAE, which normalizes MAE by the range of ground truth traffic measurements. Formally, it is expressed as:

$$\mathrm{NMAE}(t) = \frac{1}{|\mathcal{S}| \cdot |\mathcal{A}|} \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} \frac{|\widetilde{d}_a^s(t) - d_a^s(t)|}{\max_t d_a^s(t) - \min_t d_a^s(t)}. \tag{14}$$

Note that in both (13) and (14), the set of antennas $\mathcal{A}$ shall be replaced by the set of facilities or datacenters, depending on the network level at which MTD is performed.

6.3 MTD at radio access

At radio access, network slicing can provide high QoS guarantees by isolating spectrum or even dedicated antenna sites for specific mobile services [49, 50]. To fulfill this vision, resource management decisions need to be made as close as possible to the user, which in turn calls for fine-grained spatial estimates of mobile service traffic. Our first scenario for evaluation of MTD is thus one where decomposition is carried out at the antenna sector level.

6.3.1 Comparative analysis. As MTD is a completely novel problem, there is no previous solution (neither based on deep learning, nor on other approaches) that we can directly use as a benchmark. Our *modus operandi* to a comparative evaluation is thus that of assessing how the different neural network models in the literature listed in Sec. 4 perform once integrated in Microscope.

Fig. 10 gives an overview of such a comparative performance evaluation in the RAN setting, under all training methods listed in Sec. 5. We emphasize in bold the error yielded by the best model with each loss function, while the overall best performance is highlighted in red. Overall, the results are very promising. Under all

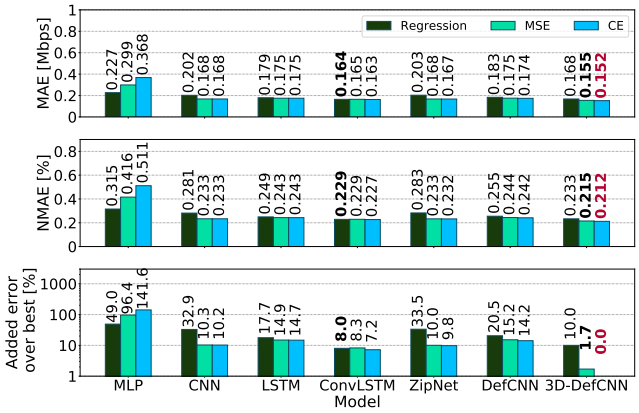

Figure 10: MTD performance in terms of MAE, NMAE, and added percent error over the best solution, across all combinations of architectures and loss functions. Results are averaged over all services and all antennas.

configurations, Microscope decomposes the total traffic recorded at one antenna into individual service-level demands with a 0.5% relative error, or lower. The corresponding MAE is always below 0.37 Mbps, which is a very reasonable inaccuracy for, *e.g.*, spectrum resource allocation to slices.

A closer look at the performance of individual schemes reveals that training with the CE loss function on traffic demand ratios leads to better estimates than using Regression or MSE. While this holds in the vast majority of cases, the improvement brought by CE over the benchmark loss functions is especially consistent for the 3D-DefCNN architecture, where it allows performance gains up to 10% in terms of MAE over the competing loss functions. This confirms that the normalization and its combination with CE indeed improve the MTD accuracy of Microscope.

As far as neural network architecture are concerned, MLP delivers the poorest MTD performance among all approaches considered, as it lacks blocks capable of extracting spatial features. As such, the added estimation error introduced by MLP with respect to 3D-DefCNN can be as high 141%. In contrast, LSTM can effectively model the temporal correlations inherent to mobile traffic, which leads to much improved results. Smaller but further improvements are then granted by convolutional architectures that can leverage spatial correlations. The performance of CNN, ConvLSTM, ZipNet, DefCNN and 3D-DefCNN are fairly aligned. Still, 3D-DefCNN yields error reductions over all other convolutional approaches that range between 7% and 33%.

The results indicate that: (*i*) both spatial and temporal features are important to solve the MTD problem; (*ii*) the spatial deformation operations help reorganizing the geographical displacements in the aggregate snapshots provided as input; (*iii*) incorporating temporal deformation into the model enables scanning of historical data with frequencies that are tailored to their importance, equivalently to a *attention-like mechanism* [60] that samples relevant information more frequently; (*iv*) a 3D-DefCNN deep learning architecture trained using a CE loss function allows Microscope to attain the highest accuracy, although performance only slightly worsen, and stay very good, under other schemes.

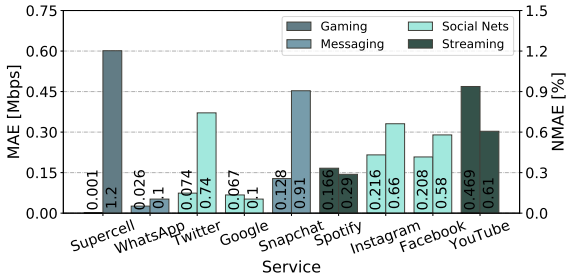

Figure 11: Microscope performance, in terms of MAE (left bars) and NMAE (right bars) for all services in $\mathcal{S}$. Results are averaged over all antennas.

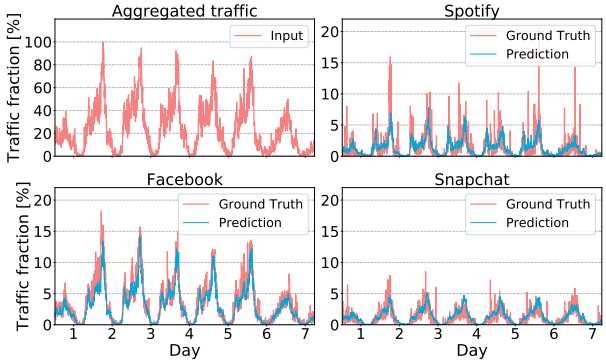

Figure 12: Example of MTD on the aggregate traffic recorded at one antenna (top left) and decompositions for three representative services by Microscope. All traffic is normalized to the aggregate activity peak.

6.3.2 Service-level performance of Microscope. We now focus on the best-performing combination of 3D-DefCNN and CE loss function, and shift our attention to the MTD performance on a per-service basis. Fig. 11 offers a breakdown of MAE (left bars) and NMAE (right bars) across the nine services in our reference set $\mathcal{S}$. There exists some apparent variability in the estimation quality across services. For instance, streaming services (*i.e.*, YouTube and Spotify), which consume a large fraction of the total traffic, are also subject to higher MAE. However, this does not necessarily lead to high relative error: in fact, their NMAE is as low as 0.61% and 0.29%, respectively. In contrast, MTD works very well for gaming traffic (*i.e.*, Supercell), due to the smaller volume of data generated by such apps. The associated NMAE is the highest recorded among all services, yet it stays at a very reasonable 1.2% figure. Overall, the MTD performance of Microscope is again remarkable, as the inference errors are well below 1% for high-demand services.

An illustrative example of the MTD quality granted by our framework is in Fig. 12, which shows the inferred time series of the demand for three representative services, *i.e.*, Spotify, Facebook, and Snapchat, based on the sole input provided by the aggregate traffic (top left subplot). The result focuses on one test week at a random single antenna. Observe that although the Spotify demand exhibits frequent fluctuations, our framework still captures well the overall traffic profile (top right). As for Facebook and Snapchat, the predicted traffic is very close to the ground truth. This confirms that Microscope yields precise MTD, irrespective of service type.

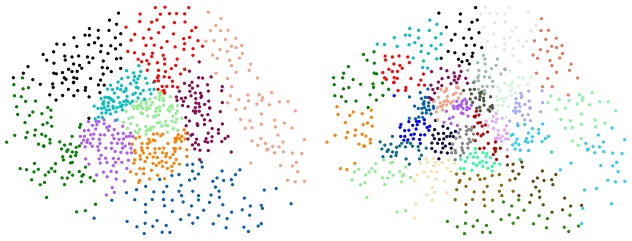

Figure 13: Assignments of antennas in the target metropolitan scenario to ten Core datacenters (left) and thirty C-RAN datacenters (right), fulfilling load balancing and latency requirements. Different colors denote clusters of antennas associated to same datacenter. Figure best viewed in color.

6.4 MTD at network datacenters

Network slicing heavily relies on the capability of the operator to dynamically orchestrate virtualized functions at the network edge and core [20]. Similarly to the radio access case, this requires efficient data-driven resource orchestration, fueled by service-level demands. In order to assess the flexibility of Microscope in helping slice management in heterogeneous edge and core network scenarios, we consider three use cases: (*i*) fifty MEC facilities deployed at the edge, aggregating traffic from 10-20 antennas each; (*ii*) thirty C-RAN datacenters, each providing MAC-layer functionalities for 20-40 antennas; (*iii*) ten core network datacenters implementing, *e.g.*, Serving Gateway (S-GW) functions [41] and accommodating traffic generated by over 50 antennas each. To decide on the associations between antennas and MEC facilities, C-RAN and core datacenters, we run the balanced graph k-partitioning algorithm proposed in [35] over the Delaunay triangulation graph [30] of the 792 antenna locations. This creates a fixed number k of antenna clusters (50, 30 or 10, for MEC, C-RAN and core datacenters, respectively) that serve comparable traffic loads, while minimizing the latency (*i.e.*, distance) with respect to associated antennas. Examples of partitions returned by the algorithm via the Karlsruhe Fast Flow Partitioning (KaFFPa) heuristic [51] are provided in Fig. 13 for the core and C-RAN datacenter scenarios.

Although the architectural settings above are arbitrary, and do not necessarily reflect the future organization of the NVF-compliant mobile network in the target region, they provide a reasonable approximation of such next-generation deployments, and allow us to investigate the performance of Microscope in diverse virtualized network scenarios. In all these cases, the geographic locations of datacenters (used by Microscope for spatial correlation inference) are the cluster centroids. We also consider different time resolutions for the management of NSaaS, from 5 mins to 1 h. The rationale is that Virtual Network Functions (VNFs) and their associated resources at datacenter level are likely to be reconfigured over longer timescales than at radio access. Hence, varying the temporal granularity offers a more complete analysis of the system.

Results are summarized in Tab. 2, using a CE loss function for all models. The minimum recorded error for each combination of network level and time resolution is highlighted in bold. The key takeaway is that Microscope still performs very well, and MTD allows reconstructing service-level demands with best relative errors well below 2% in all NSaaS network management settings.

Table 2: MAE (top, in MB per NSaaS management interval) and NMAE (bottom, in %) returned by all models over NSaaS management intervals from 5 minutes to 1 hour. Each element reports the mean MTD error at core datacenters (left), C-RAN datacenters (middle) and MEC facilities (right).

Model	5 mins	10 mins	30 mins	1 h
MLP	116/53/38	213/96/69	602/263/175	5022/673/401
CNN	152/64/43	290/119/79	842/321/206	1617/589/306
LSTM	**106**/52/38	**191**/93/67	**483**/236/163	**895**/**414**/**283**
ConvLSTM	129/55/39	268/104/71	837/319/199	1026/611/345
ZipNet	149/60/41	281/110/75	699/262/185	1313/524/344
DefCNN	127/53/37	228/96/67	632/244/156	1315/491/295
3D-DefCNN	124/**49**/**34**	231/**89**/**62**	696/**230**/**155**	1194/451/292

Model	5 mins	10 mins	30 mins	1 h
MLP	1.99/1.58/1.23	1.96/1.58/1.62	1.91/1.84/1.61	8.37/2.56/1.99
CNN	2.61/1.90/1.38	2.67/1.97/1.86	2.67/2.25/1.88	2.70/2.24/1.92
LSTM	**1.82**/1.56/1.21	**1.76**/1.54/1.58	**1.53**/1.65/1.50	**1.49**/**1.57**/**1.40**
ConvLSTM	2.22/1.64/1.26	2.47/1.71/1.68	2.65/2.23/1.82	2.71/2.32/1.71
ZipNet	2.56/1.77/1.31	2.59/1.82/1.76	2.22/1.84/1.69	2.19/1.99/1.70
DefCNN	2.18/1.58/1.20	2.10/1.59/1.60	2.00/1.71/1.43	2.19/1.87/1.46
3D-DefCNN	2.13/**1.47**/**1.08**	2.12/**1.48**/**1.47**	2.21/**1.61**/**1.42**	1.99/1.71/1.45

Again, 3D-DefCNN outperforms all other architectures as long as MTD is run at high frequency (*i.e.*, 5 to 30 minutes) and closer to the user (*i.e.*, at MEC facilities and C-RAN datacenters). We remark that these are the most challenging conditions for the estimation of mobile service traffic. Instead, when mobile traffic is accumulated in large volumes, by considering hourly aggregates or demands at network core datacenters, LSTM yields the lowest errors. In this case, coarse temporal granularities lead to time series that are more regular, and generally easier to decompose. Similarly, considering traffic combined at a relatively small number of core datacenters diminishes the impact of spatial correlations between locations. Under these settings, the complexity of 3D-DefCNN becomes unnecessary, whereas LSTM thrives by avoiding looking for complicate interactions that are in fact absent in the input data. For the same reason, the performance of all CNN-based models improves as we move from the core to the edge of the network, where the impact of spatial correlations increases.

Overall, our analysis indicates that: (*i*) MTD retains high accuracy and is a viable approach to service-level demand estimation also for NSaaS at the mobile network edge and core; (*ii*) different neural network models shall be adopted within Microscope to ensure the best performance at different network locations, as architectures capable of extracting spatial features are important close to the radio access, whereas temporal correlations become more critical in the network core and at longer management timescales.

6.5 Complexity Analysis

We also evaluate the complexity of models in terms of floating point operations (FLOPs) per inference instance, a metric frequently employed with neural networks [37]. The number of FLOPs is computed by counting the number of mathematical operation or assignments that involve floating-point numbers. As shown in Fig. 14, all models that include convolution operations (*i.e.*, CNN, ConvLSTM, ZipNet, DefCNN, and 3D-DefCNN) are sensitive to the spatial granularity of the input data, hence their complexity varies significantly with the mobile network level. In contrast, MLP and LSTM yield a complexity that is not affected by the input size.

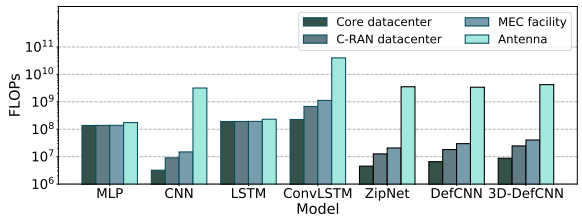

Figure 14: Complexity (measured in FLOPs) of all evaluated neural network models across different network levels.

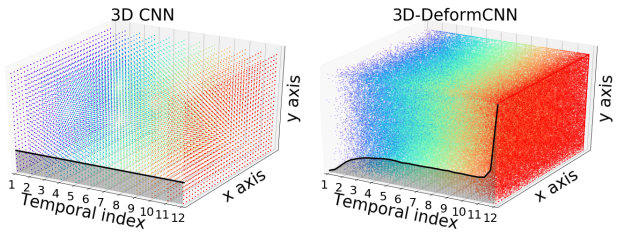

Figure 15: Example of spatiotemporal distribution of positions in one tensor input visited by the filter of a traditional 3D convolutional layer (left), and by the filter of the proposed 3D deformable convolutional layer (right).

Interestingly, although LSTM yields the best accuracy at core network datacenters, as indicated in Tab. 4, it also entails the highest complexity in that scenario, exceeding that of 3D-DefCNN by more than one order of magnitude. LSTM also has very high complexity when MTD is run at C-RAN datacenters or MEC facilities. Conversely, the computational requirements of CNN-based models surpass those of LSTM only for antenna-level MTD.

Moreover, Fig. 14 makes it clear that deformable operations do not introduce significant additional complexity compared to plain CNN, despite the important advantage in terms of accuracy in the most testing scenarios that require fast management of NSaaS resources located close to the user. The reason is that a 3D deformable convolution effectively enables the scanning of historical data with frequencies that are dependent on their importance, equivalently to an attention-like mechanism that samples relevant information more frequently [60]. We illustrate the effect in Fig. 15, which compares the spatiotemporal distribution of the elements of one tensor input (*i.e.*, a set of consecutive aggregate traffic snapshots) visited by a legacy 3D convolutional ingress layer (employed, *e.g.*, by ZipNet and DefCNN) and by our novel 3D deformable convolutional ingress layer (adopted by 3D-DefCNN). The plots also show the density of sampled points projected onto the temporal dimension (*i.e.*, $x = 0$ surface) as black solid lines subtending a shaded area.

The structure shown on the left is not flexible, leading to a regular sampling in space and time, which results into a uniform distribution of points over all input matrices. Instead, the 3D deformable convolutional layer learns which positions of the input tensor are the most relevant: as a result, it samples elements in each input matrix non uniformly, and with a higher density of samples in recent input matrices. This way, the layer can better exploit more relevant information in fresher snapshots. While Fig. 15 just provides a visual example for one specific tensor input, the same behavior

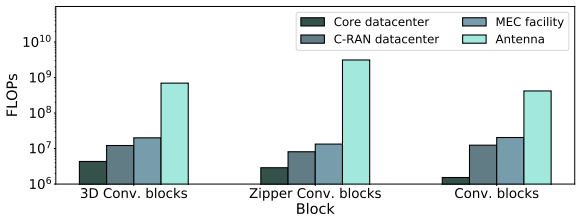

Figure 16: Complexity (measured in FLOPs) of each block in the 3D-DefCNN model across different network levels.

is observed at the input layer of 3D-DefCNNs trained in all network scenarios and NSaaS management timescales. As a result, 3D-DefCNN models sample *better*, and not *more often* the input, which explains the improved performance at equivalent complexity.

Further details on the 3D-DefCNN computational cost are provided in Fig. 16, which breaks down the complexity associated to each block of the model in Fig. 8. The cost ratio of the three components remains fairly comparable across all network settings, proving that a 3D deformable convolution does not entail complexity surges in any scenario, and thus corroborating the considerations above.

Overall, 3D-DefCNN requires around 4×10^9 FLOPs per inference instance for antenna-level MTD, which are easily handled by modern CPUs: *e.g.*, an Intel Core i7 980 XE can execute 1.076×10^{11} FLOPs per second, and can thus fully support real-time MTD.

7 NSaaS MANAGEMENT USE CASES

We complete our evaluation of the performance of Microscope by assessing the viability of MTD in practical case studies of NSaaS resource management. Specifically, we take the perspective of the mobile network operator, and assess the incurred costs when Microscope is used to determine the capacity allocated to each network slice, solely based on aggregate traffic information.

7.1 Datacenter resource management

At network datacenters, resource management costs directly stem from estimation errors in the decomposed demands, which may entail Service-Level Agreements (SLAs) violations (if the per-service traffic estimate is below the reference) or overprovisioning of unnecessary capacity to specific slices (if the same estimate is above the reference). To assess the quality of MTD in this context, we re-train Microscope with a recently proposed loss function that specifically aims at minimizing monetary costs for network operators [5], hence fully integrating our framework into NSaaS management operations. The loss function is:

$$\mathrm{L}(t) = \frac{1}{|\mathcal{S}| \cdot |\mathcal{A}|} \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} \mathcal{L}(\Delta d_a^s(t)), \tag{15}$$

where

$$\mathcal{L}(\Delta d_a^s(t)) = \begin{cases} \alpha - \epsilon \cdot \Delta d_a^s(t), & \Delta d_a^s(t) \leq 0 \\ \alpha - \frac{1}{\epsilon} \Delta d_a^s(t), & 0 < \Delta d_a^s(t) \leq \alpha\epsilon \\ \Delta d_a^s(t) - \alpha\epsilon, & \Delta d_a^s(t) > \alpha\epsilon. \end{cases} \tag{16}$$

Here $\Delta d_a^s(t) = \widetilde{d}_a^s(t) - d_a^s(t)$, α controls the cost of an SLA violation, and ϵ is a small constant. The expression accounts for the cost of SLA violations (a fixed fee paid when $\Delta d_a^s(t) \leq \alpha\epsilon$) and overprovisioning (growing as additional resources are erroneously reserved, when

Table 3: Total SLA violation cost and overprovisioning cost determined by MTD at different network levels.

Use case	SLA violation [MB/s (%)]	Overprovisioning [MB/s (%)]
MEC facility	247.95 (136.31)	203.38 (111.80)
C-RAN datacenter	25.82 (14.20)	106.59 (58.60)
Core datacenter	15.54 (8.55)	61.71 (33.92)

Table 4: Mean additional costs of antenna-level MTD.

Throughput	Subcarriers	Spectrum cost	CPU time
6.114 Mbps	110	3 MHz	7.5%

$\Delta d_a^s(t) > \alpha\epsilon$), so that Microscope can perform MTD by trying to balance them. We configure $\alpha = 1$ and $\epsilon = 0.01$, as suggested in [5].

We test Microscope with this configuration in three case studies: (*i*) MTD at MEC facilities on traffic aggregated at every 30 minutes; (*ii*) MTD at C-RAN datacenters on traffic aggregated at every 30 minutes; and (*iii*) MTD at core datacenters on traffic aggregated at every hour. Exclusively in the last scenario, we choose LSTM as the neural network architecture, according to the results in Sec.6.

The costs incurred by the operator in the datacenter scenarios (*i*)–(*iii*) above are listed in Tab. 3. The table expresses costs in MB/s, which can then be translated into actual monetary values based on the price of the technology implementing such capacity at each network level. For overprovisioning, the cost maps to proper additional MB/s of allocated capacity beyond what strictly required; for SLA violations, each infringement has the same cost as allocating additional capacity to cover α times the peak demand, as per (15). In both cases, results are reported as the total over all MEC (respectively, C-RAN and network core) nodes in the target region.

At C-RAN and core datacenters, Microscope carries percent costs in the range from 8% to 58%, computed with respect to the true demand. These are in fact comparable to the equivalent costs for capacity allocation to network slices at the same network levels, incurred when the operator has perfect knowledge of the traffic demand associated to each mobile service. Indeed, using a state-of-the-art one-step predictor in these conditions yields costs up to 30% for SLA violations and up to 18% for overprovisioning [5]. When pushing Microscope at MEC facilities, performance degrade sensibly: strong fluctuations in the traffic make a MTD-based estimation of the per-service demand less suitable to capacity allocation.

7.2 Radio access resource management

When MTD is performed at antenna level, a relevant metric are the extra subcarriers/spectrum resources required to support the unnecessary capacity overprovisioned at each antenna to every slice [43]. We can also quantify those resources in terms of CPU time, based on experimental models obtained with open LTE stacks [18]. Tab. 4 reports Microscope results under such models. When MTD is performed at antenna level, just 6 Mbps of additional throughput are needed per antenna, which yields a 3 MHz spectrum cost and requires 7.5% additional CPU time with respect to the case where perfect knowledge of service traffic is available.

These results, jointly with those for network datacenter use cases above, let us conclude that MTD can be a viable low-cost approach to service-level demand estimation in practical NSaaS management cases, where it enables effective network resource allocations.

8 RELATED WORK

Relevant to our study are works on (*i*) mobile traffic analysis, and (*ii*) machine learning solutions for time series decomposition.

Mobile traffic analytics are becoming increasingly vital to mobile operators and a number of directions are being explored. For instance, Shafiq *et al.* carried out seminal work on the geospatial correlations of traffic volume and application usage in cellular networks [33, 53]. Wang *et al.* proposed models that combine location information, time dimension, and the traffic frequency spectrum, to extract traffic patterns in urban settings [57]. Furno *et al.* investigated traffic signatures to classify mobile demands across 10 different cities [14]. Marquez *et al.* revealed strong heterogeneity in the demand of mobile services, by employing correlation and clustering [36]. Traffic demand in narrowly localized regions was inferred from coarse aggregates using a neural network in [63]. However, to the best of our knowledge, the problem of mobile network traffic decomposition (MTD) that we tackle in this work has not been addressed to date.

Time series decomposition is traditionally posed as a single-channel blind source separation problem and solved using Independent Component Analysis (ICA) [23]. However, this approach only works on short sequences. Machine learning tools implementing additive factorial hidden Markov models (AFHMMs) circumvent this problem in the context of energy consumption disaggregation [66]. More recently, CNNs were proposed as alternatives that perform sequence-to-point learning using a sliding window approach on very long time series [65]. Both these approaches are limited to single time series decomposition and, unlike the proposed 3D-DefCNN that we propose for use with Microscope, they do not exploit spatiotemporal correlations in the input data.

9 CONCLUSIONS

We introduced Microscope, a dedicated framework for aggregate Mobile Traffic Decomposition (MTD) into service-level demands, intended to assist resource allocation to network slices in NSaaS environments. The framework feeds suitably transformed traffic data to a flexible deep learning model, whose architecture can be adapted to the NSaaS management location or timescale. Performance evaluations with measurement data demonstrate that Microscope provides accurate traffic inference in real-time, and we show that a resource allocation based on decomposition yields affordable costs for the operator. As a result, MTD via Microscope provides a means to complement and limit the need for extensive deep packet inspection (DPI) in traffic collected at different levels of the network. As such, our approach has the potential to practically solve computationally intensive traffic analytics, which is essential to agile provisioning of resource in 5G mobile networks.

Acknowledgments

The authors would like to thank the HPC@POLITO academic computing initiative (http://hpc.polito.it) for providing the computational resources that supported this research. Paul Patras acknowledges the support received from Cisco Systems, Inc. through the University Research Program Fund, gift no. 2019-197006. This work was supported by the ANR CANCAN project (ANR-18-CE25-0011). The authors are grateful for the reviewers' constructive feedback, and for the shepherd's guidance during the revision process.

REFERENCES

[1] 3GPP TS Group Services and System Aspects; Telecommunication management. 2018. Study on management and orchestration of network slicing for next generation network. TR 28.801 V15.1.0.

[2] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. TensorFlow: A System for Large-Scale Machine Learning.. In *OSDI*, Vol. 16. 265–283.

[3] Arjun Anand, Gustavo de Veciana, and Sanjay Shakkottai. 2018. Joint Scheduling of URLLC and eMBB Traffic in 5G Wireless Networks. *Proc. IEEE INFOCOM* (2018), 1970–1978.

[4] Gianni Barlacchi, Marco De Nadai, Roberto Larcher, Antonio Casella, Cristiana Chitic, Giovanni Torrisi, Fabrizio Antonelli, Alessandro Vespignani, Alex Pentland, and Bruno Lepri. 2015. A multi-source dataset of urban life in the city of Milan and the Province of Trentino. *Scientific Data* (2015).

[5] Dario Bega, Marco Gramaglia, Marco Fiore, Albert Banchs, and Xavier Costa-Perez. 2019. DeepCog: Cognitive Network Management in Sliced 5G Networks with Deep Learning. In *Proc. IEEE INFOCOM*.

[6] C. Bishop. 2006. *Pattern Recognition and Machine Learning*. Springer.

[7] Hanna Bogucka, Paweł Kryszkiewicz, and Adrian Kliks. 2015. Dynamic spectrum aggregation for future 5G communications. *IEEE Comm. Mag.* 53, 5 (2015), 35–43.

[8] C. Zhang *et al.* 2019. Deep Learning in Mobile and Wireless Networking: A Survey. *IEEE Comms Surveys & Tutorials* (2019).

[9] L. M. Contreras and D. R. López. 2018. A Network Service Provider Perspective on Network Slicing. *IEEE Softwarization* (Jan 2018).

[10] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. 2017. Deformable Convolutional Networks. In *IEEE International Conference on Computer Vision (ICCV)*. 764–773.

[11] Hao Dong, Akara Supratak, Luo Mai, Fangde Liu, Axel Oehmichen, Simiao Yu, and Yike Guo. 2017. TensorLayer: A Versatile Library for Efficient Deep Learning Development. In *Proc. ACM on Multimedia Conference* (Mountain View, California, USA). 1201–1204.

[12] ETSI. 2018. Open Source MANO Release FIVE Technical Overview.

[13] A. Fiessler, C. Lorenz, S. Hager, B. Scheuermann, and A. W. Moore. 2017. HyPaFilter+: Enhanced Hybrid Packet Filtering Using Hardware Assisted Classification and Header Space Analysis. *IEEE/ACM Transactions on Networking* 25, 6 (2017), 3655–3669.

[14] A. Furno, M. Fiore, R. Stanica, C. Ziemlicki, and Z. Smoreda. 2017. A Tale of Ten Cities: Characterizing Signatures of Mobile Traffic in Urban Areas. *IEEE Transactions on Mobile Computing* 16, 10 (Oct 2017), 2682–2696. https://doi.org/10.1109/TMC.2016.2637901

[15] Pavel Golik, Patrick Doetsch, and Hermann Ney. 2013. Cross-entropy vs. squared error training: a theoretical and experimental comparison.. In *Interspeech*, Vol. 13.

[16] Marta C. Gonzalez, Cesar A. Hidalgo, and Albert-Laszlo Barabasi. 2008. Understanding individual human mobility patterns. *Nature* 453, 7196 (June 2008), 779–782. https://doi.org/10.1038/nature06958

[17] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press.

[18] Francesco Gringoli, Paul Patras, Carlos Donato, Pablo Serrano, and Yan Grunenberger. 2018. Performance Assessment of Open Software Platforms for 5G Prototyping. *IEEE Wireless Communications* 25, 5 (2018), 10–15.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*.

[20] Peter Hedman. 2016. NGMN 5G Requirements & Architecture WS End-to-End Architecture – Description of Network Slicing Concept.

[21] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[22] Z. Hu, L. Zhu, J. Heidemann, A. Mankin, D. Wessels, and P. Hoffman. 2016. *Specification for DNS over Transport Layer Security (TLS).* RFC 7858.

[23] Aapo Hyvärinen, Juha Karhunen, and Erkki Oja. 2004. *Independent component analysis*. Vol. 46. John Wiley & Sons.

[24] IHS Economics/Technology. 2017. The 5G economy: How 5G technology will contribute to the global economy. (2017).

[25] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*. 448–456.

[26] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 2013. 3D convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence* 35, 1 (2013), 221–231.

[27] Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proc. International Conference on Learning Representations*.

[28] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet classification with deep convolutional neural networks. In *Proc. NIPS*.

[29] Harold W Kuhn. 1955. The Hungarian method for the assignment problem. *Naval Research Logistics (NRL)* 2, 1-2 (1955), 83–97.

[30] Der-Tsai Lee and Bruce J Schachter. 1980. Two algorithms for constructing a Delaunay triangulation. *International Journal of Computer & Information Sciences* 9, 3 (1980), 219–242.

[31] Hao Li, Zheng Xu, Gavin Taylor, and Tom Goldstein. 2018. Visualizing the Loss Landscape of Neural Nets. *NIPS* (2018).

[32] R. Li, Z. Zhao, X. Zhou, J. Palicot, and H. Zhang. 2014. The prediction analysis of cellular radio access network traffic: From entropy theory to networking practice. *IEEE Communications Magazine* 52, 6 (2014), 234–240.

[33] M. Z. Shafiq *et al.* 2012. Characterizing geospatial dynamics of application usage in a 3G cellular data network. In *Proc. IEEE INFOCOM*. 1341–1349.

[34] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *Proc. ICML*, Vol. 30. 3.

[35] Cristina Marquez, Marco Gramaglia, Marco Fiore, Albert Banchs, and Xavier Costa-Perez. 2018. How Should I Slice My Network?: A Multi-Service Empirical Evaluation of Resource Sharing Efficiency. In *Proc. ACM MobiCom*.

[36] Cristina Marquez, Marco Gramaglia, Marco Fiore, Albert Banchs, Cezary Ziemlicki, and Zbigniew Smoreda. 2017. Not All Apps Are Created Equal: Analysis of Spatiotemporal Heterogeneity in Nationwide Mobile Service Usage. In *Proc. ACM CoNEXT*.

[37] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. 2017. Pruning convolutional neural networks for resource efficient inference. In *ICLR*.

[38] Kevin P Murphy. 2012. *Machine learning: a probabilistic perspective*. MIT press.

[39] Navid Nikaein. 2015. Processing Radio Access Network Functions in the Cloud: Critical Issues and Modeling. In *Proc. Intl Workshop on Mobile Cloud Computing and Services*. 36–43.

[40] Official Journal of the European Union. 2016. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation).

[41] ONF. 2019. Converged Multi-Access and Core (COMAC). https://www.opennetworking.org/comac/

[42] P. Orosz, T. Tóthfalusi, and P. Varga. 2019. FPGA-Assisted DPI Systems: 100 Gbit/s and Beyond. *IEEE Communications Surveys & Tutorials* 21, 2 (2019), 2015–2040.

[43] Prashant Panigrahi. 2015. How to Calculate LTE Data Rate – Downlink Throughput. http://www.3glteinfo.com/lte-data-rate-throughput/. [Online; accessed Feb-2019].

[44] Utpal Paul, Anand Prabhu Subramanian, Milind M. Buddhikot, and Samir R. Das. 2011. Understanding traffic dynamics in cellular data networks. In *Proc. IEEE INFOCOM*.

[45] H Qi, Z Zhang, B Xiao, H Hu, B Cheng, Y Wei, and J Dai. 2017. Deformable convolutional networks–COCO detection and segmentation challenge 2017 entry. In *ICCV COCO Challenge Workshop*.

[46] DA Rajon and WE Bolch. 2003. Marching cube algorithm: review and trilinear interpolation adaptation for image-based dosimetric models. *Computerized Medical Imaging and Graphics* 27, 5 (2003).

[47] Eric Rescorla, Kazuho Oku, Nick Sullivan, and Christopher Wood. 2018. *Encrypted Server Name Indication for TLS 1.3.* Internet-Draft draft-ietf-tls-esni-02. IETF Secretariat.

[48] S. Rezaei and X. Liu. 2019. Deep Learning for Encrypted Traffic Classification: An Overview. *IEEE Comm. Mag.* 57, 5 (May 2019), 76–81.

[49] P. Rost, C. Mannweiler, D. S. Michalopoulos, C. Sartori, V. Sciancalepore, N. Sastry, O. Holland, S. Tayade, B. Han, D. Bega, D. Aziz, and H. Bakker. 2017. Network Slicing to Enable Scalability and Flexibility in 5G Mobile Networks. *IEEE Communications Magazine* 55, 5 (May 2017), 72–79. https://doi.org/10.1109/MCOM.2017.1600920

[50] O. Sallent, J. Perez-Romero, R. Ferrus, and R. Agusti. 2017. On Radio Access Network Slicing from a Radio Resource Management Perspective. *IEEE Wireless Communications* 24, 5 (October 2017), 166–174. https://doi.org/10.1109/MWC.2017.1600220WC

[51] Peter Sanders and Christian Schulz. 2013. Think Locally, Act Globally: Highly Balanced Graph Partitioning. In *Proc. International Symposium on Experimental Algorithms (SEA) (LNCS, Vol. 7933)*. Springer, 164–175.

[52] V. Sciancalepore, K. Samdanis, X. Costa-Perez, D. Bega, M. Gramaglia, and A. Banchs. 2017. Mobile traffic forecasting for maximizing 5G network slicing resource utilization. In *Proc. IEEE INFOCOM*.

[53] M. Zubair Shafiq, Lusheng Ji, Alex X. Liu, and Jia Wang. 2011. Characterizing and Modeling Internet Traffic Dynamics of Cellular Devices. In *Proc. SIGMETRICS*.

[54] T. Taleb, A. Ksentini, and R. Jantti. 2016. 'Anything as a Service' for 5G Mobile Systems. *IEEE Network* 30, 6 (Nov. 2016), 84–91.

[55] Martino Trevisan, Danilo Giordano, Idilio Drago, Marco Mellia, and Maurizio Munafo. 2018. Five Years at the Edge: Watching Internet from the ISP Network. In *Proc. ACM CoNEXT '18* (Heraklion, Greece). 1–12.

[56] Andreas Veit, Michael J Wilber, and Serge Belongie. 2016. Residual networks behave like ensembles of relatively shallow networks. In *NIPS*.

[57] Huandong Wang, Fengli Xu, Yong Li, Pengyu Zhang, and Depeng Jin. 2015. Understanding Mobile Traffic Patterns of Large Scale Cellular Towers in Urban Environment. In *Proc. ACM IMC*.

[58] Cort J Willmott and Kenji Matsuura. 2005. Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model

performance. *Climate research* 30, 1 (2005).
[59] SHI Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. 2015. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Proc. NIPS*.
[60] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. 2018. Generative Image Inpainting with Contextual Attention. In *Proc. IEEE CVPR*.
[61] Chaoyun Zhang, Marco Fiore, Iain Murray, and Paul Patras. 2019. CloudLSTM: A Recurrent Neural Model for Spatiotemporal Point-cloud Stream Forecasting. *preprint arXiv:1907.12410* (2019).
[62] Chaoyun Zhang, Marco Fiore, and Paul Patras. 2019. Multi-Service Mobile Traffic Forecasting via Convolutional Long Short-Term Memories. In *IEEE International Symposium on Measurements & Networking (M&N)*.
[63] Chaoyun Zhang, Xi Ouyang, and Paul Patras. 2017. ZipNet-GAN: Inferring Fine-grained Mobile Traffic Patterns via a Generative Adversarial Neural Network. In *Proc. ACM CoNEXT*. 363–375.
[64] Chaoyun Zhang and Paul Patras. 2018. Long-Term Mobile Traffic Forecasting Using Deep Spatio-Temporal Neural Networks. In *Proc. ACM MobiHoc*.
[65] Chaoyun Zhang, Mingjun Zhong, Zongzuo Wang, Nigel Goddard, and Charles Sutton. 2018. Sequence-to-point learning with neural networks for nonintrusive load monitoring. In *AAAI*.
[66] Mingjun Zhong, Nigel Goddard, and Charles Sutton. 2014. Signal aggregate constraints in additive factorial HMMs, with application to energy disaggregation. In *Advances in Neural Information Processing Systems*.
[67] Jian Zhu, Leyuan Fang, and Pedram Ghamisi. 2018. Deformable Convolutional Neural Networks for Hyperspectral Image Classification. *IEEE Geos. & Remote Sens. Let.* (2018).